嵌入式系统

张武明

第十讲　Cortex-M3架构和指令系统

§1嵌入式系统的应用领域

§2 ARM公司简介

§3　Cortex-M3内核

§4　指令系统

§5　嵌入式编程

§10.1 嵌入式系统的应用领域

- 国防武器设备。
- 通信信息设备。
- 过程控制。
- 智能仪器。
- 消费产品。
- 生物微电子技术。

嵌入式系统的出现和兴起

- 出现：20世纪60年代以晶体管、磁芯存储为基础的计算机开始用于航空等军用领域。
  - 第一台机载专用数字计算机是奥托内蒂克斯公司为美国海军舰载轰炸机“民团团员”号研制的多功能数字分析器(Verdan)。
  - 同时嵌入式计算机开始应用于工业控制。1962年一个美国乙烯厂实现了工业装置中的第一个直接数字控制(DDC)。

嵌入式系统的出现和兴起

- 兴起：在1965～1970年，当时计算机已开始采用集成电路，即第三代计算机。在军事、航空航天领域、工业控制的需求推动下。
  - 第一次使用机载数字计算机控制的是1965年发射的Gemini3号，第一次通过容错来提高可靠性是1968年的阿波罗4号、土星5号。
  - 1963年DEC公司推出PDP8并发展成PDP11系列，成为工业生产集中控制的主力军。
  - 在军用领域中，为了可靠和满足体积、重量的严格要求，还需为各个武器系统设计五花八门的专用的嵌入式计算机系统。

嵌入式系统开始走向繁荣

- 嵌入式系统大发展是在微处理器问世之后
  - 1973年至1977年间各厂家推出了许多8位的微处理器，包括Intel 8080/8085，Motorola 的6800/6802，Zilog的Z80和Rockwell的6502。
  - 微处理器不单用来组成微型计算机，而且用来制造仪器仪表、医疗设备、机器人、家用电器等嵌入式系统。
  - 仅8085/Z80微处理器的销售就超过7亿片,其中大部分是用于嵌入式工业控制应用。

嵌入式系统的应用领域

- 嵌入式系统广泛地应用于消费电子、通信、汽车、国防、航空航天、工业控制、仪表、办公自动化等领域。
- 据欧盟的统计：
  - 2003年全球大概有80亿片嵌入式微处理器，到2010年，预计会达到160亿片，地球上的人平均拥有3个嵌入式微处理器；
  - 在航空电子中，嵌入式软件的开发成本占整个飞机研制成本的50%；对于汽车工业，汽车电子在整车价值中的比例逐年提高，将从1997年的20%提升到2010年的33-40%；
  - 消费电子数量越来越大，据预测，到2010年，仅数字家庭在美国的销售额就将达到2000亿欧元。

嵌入式系统的应用领域

A “short list” of embedded systems

- Anti-lock brakes
- Auto-focus cameras
- Automatic teller machines
- Automatic toll systems
- Automatic transmission
- Avionic systems
- Battery chargers
- Camcorders
- Cell phones
- Cell-phone base stations
- Cordless phones
- Cruise control
- Curbside check-in systems
- Digital cameras
- Disk drives
- Electronic card readers
- Electronic instruments
- Electronic toys/games
- Factory control
- Fax machines
- Fingerprint identifiers
- Home security systems
- Life-support systems
- Medical testing systems
- Modems
- MPEG decoders
- Network cards
- Network switches/routers
- On-board navigation
- Pagers
- Photocopiers
- Point-of-sale systems
- Portable video games
- Printers
- Satellite phones
- Scanners
- Smart ovens/dishwashers
- Speech recognizers
- Stereo systems
- Teleconferencing systems
- Televisions
- Temperature controllers
- Theft tracking systems
- TV set-top boxes
- VCR’s, DVD players
- Video game consoles
- Video phones
- Washers and dryers

嵌入式系统的应用领域

- 消费电子领域
  - 随着技术的发展，消费电子产品正向数字化和网络化方向发展。
  - 高清晰度数字电视将代替传统的模拟电视。
  - 数码相机将代替传统的胶片相机。
  - 固定电话今后会被IP电话所替代。
  - 各种家用电器（电视机、冰箱、微波炉、电话等）将通过家庭通信、控制中心与Internet连接，实现远程控制、信息交互、网上娱乐、远程医疗和远程教育等。转变为智能网络家电，还可以实现远程医疗，远程教育等。

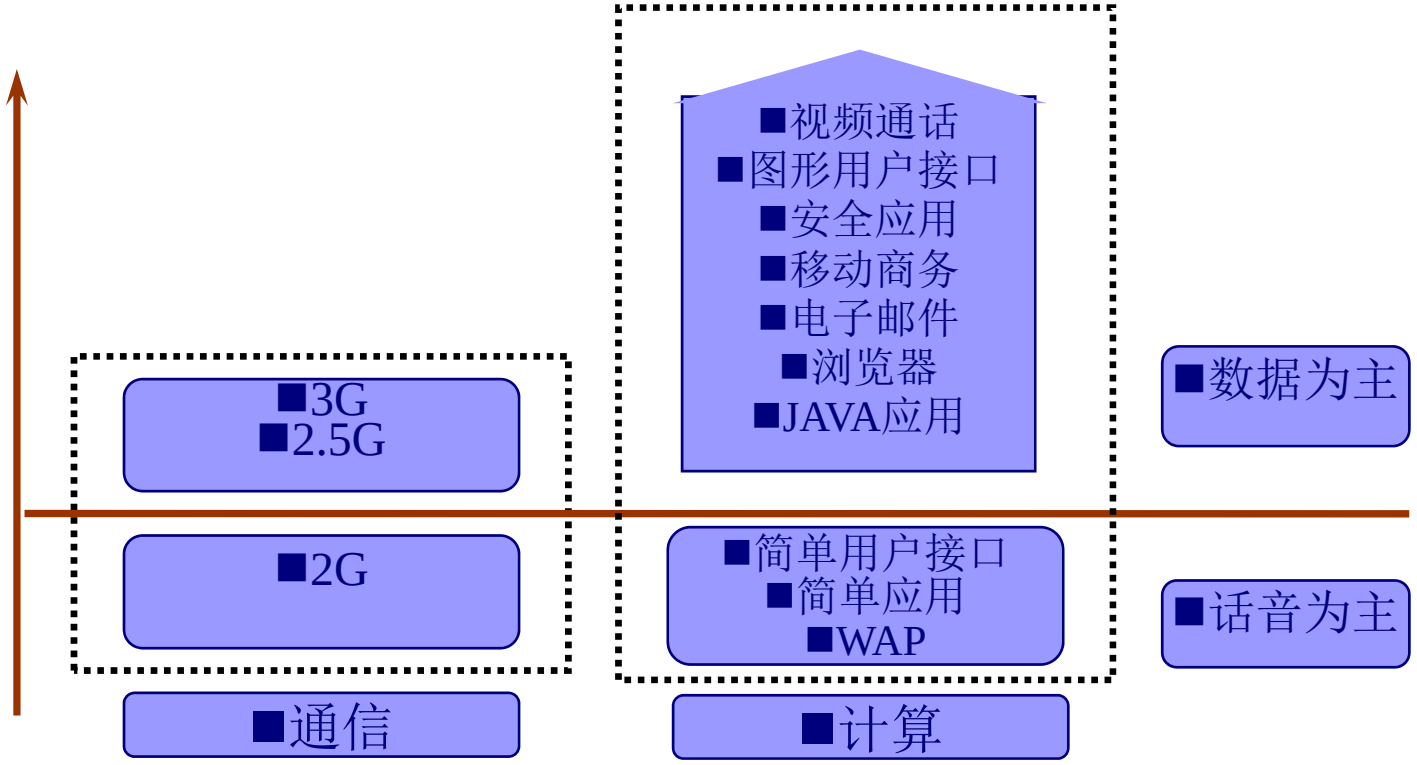
嵌入式系统的应用领域

- 工控、汽车电子、仿真、医疗仪器等
  - 随着工业、汽车、医疗卫生等各部门对智能控制需求的不断增长，需要对设备进行智能化、数字化改造，为嵌入式系统提供了很大的市场。
  - 就汽车电子系统而言，目前的大多数高档轿车每辆拥有约50个嵌入式微处理器。如BMW 7系列轿车，则平均安装有63个嵌入式微处理器。
  - 据预测，21世纪初美国接入Internet的汽车将有一亿辆。IC Insights报道2001年车载计算系统的市场规模是30亿美元，而2004年将达到46亿美元，届时这些系统将成为所有新型轿车的标准设备。

嵌入式系统的应用领域

- 通信领域
  - 通信领域大量应用嵌入式系统，主要包括程控交换机、路由器、IP交换机、传输设备等。
  - 据预测，由于互联的需要，特别是宽带网络的发展，将会出现各种网络设备如：ADSL Modem/Router等，其数量将远远高于传统的网络设备。
  - 它们基于32位的嵌入式系统、价格低廉，将为企业、家庭提供更为廉价的、方便的、多样的网络方案。就宽带上网的网络设备ADSL Router而言，国外现在每月需要600K的数量。

嵌入式系统的应用领域

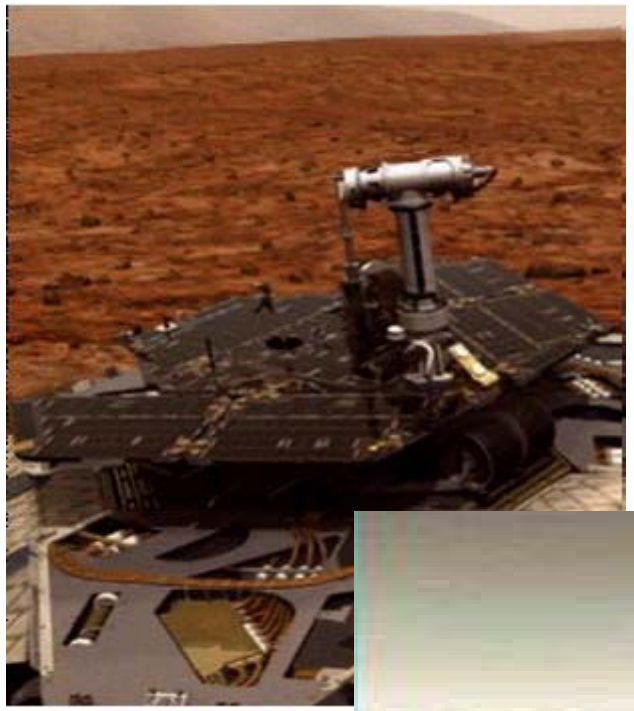


手机发展趋势

基于RTLinux的仿人机器人

高 48 cm
重: 6 kg
灵活性： 20 DOF
操作系统: RT-Linux
接口形式: USB 1.0 (12Mbps)
响应周期: 1ms
能源： DC24V x 6.2A (150W)
制造： 富士通

2004年“勇气号”再次登陆火星



远在火星的自动车

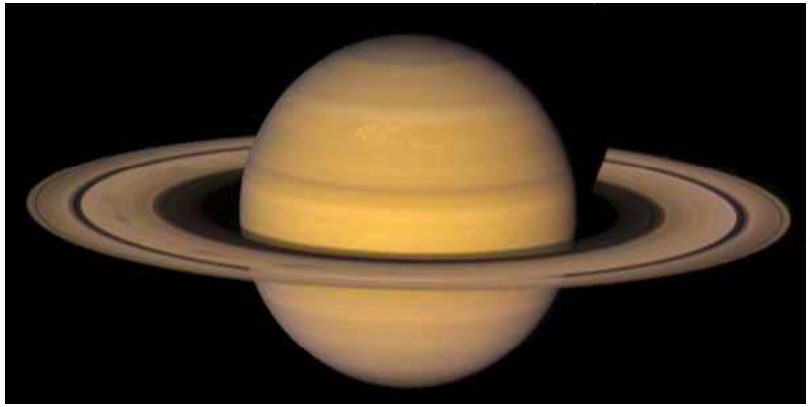
■火星与地球，这一对在星空中遥遥相望的“兄弟”，迎来6万年来“最亲密的接触”，在2003年8月27日这一天，火星距离地球最近达到55756622(5千多万)公里。

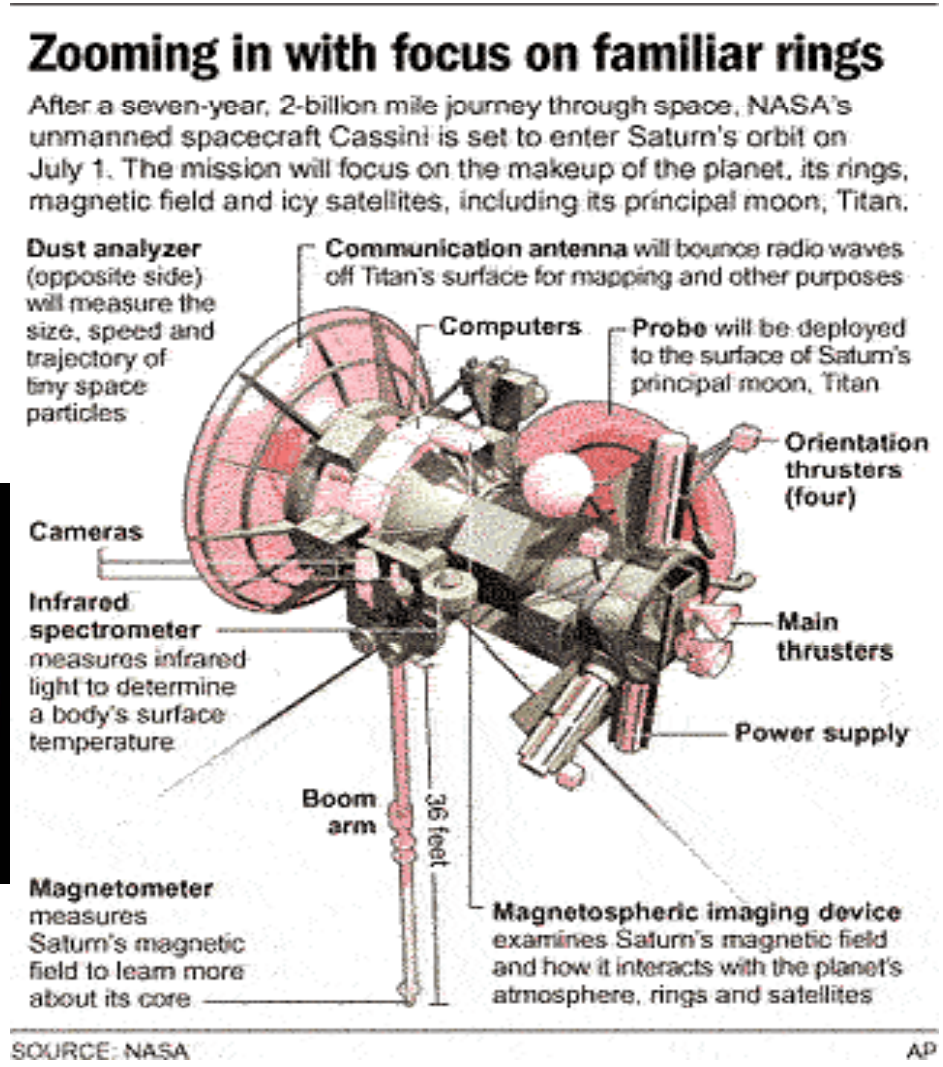
■勇气号

■面对6万年才有一次的机会，科学家们积极行动起来———从6月开始，先后有欧洲的“火星快车”、美国“勇气号”和“机遇号”等三颗火星探测器飞往火星，而日本一颗本已在太空“迷失方向”的火星探测器也在关键时刻及时“醒”来，开始了久违的火星之旅。

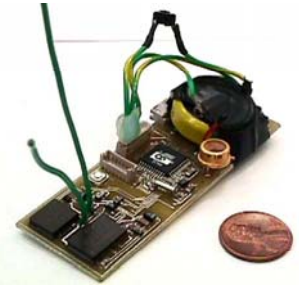
■土星探测

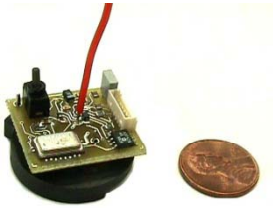
■自1997年10月15日发射以来，经历了7年35亿公里航程的卡西尼号太空船在2004年7月1日10时30分进入土星轨道，开始进行人类有史以来对土星及其31颗已知卫星最详尽的探测。

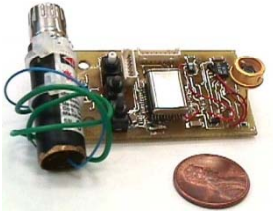


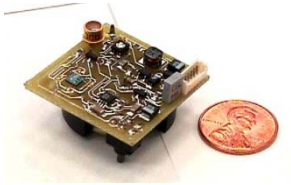


Smart Dust



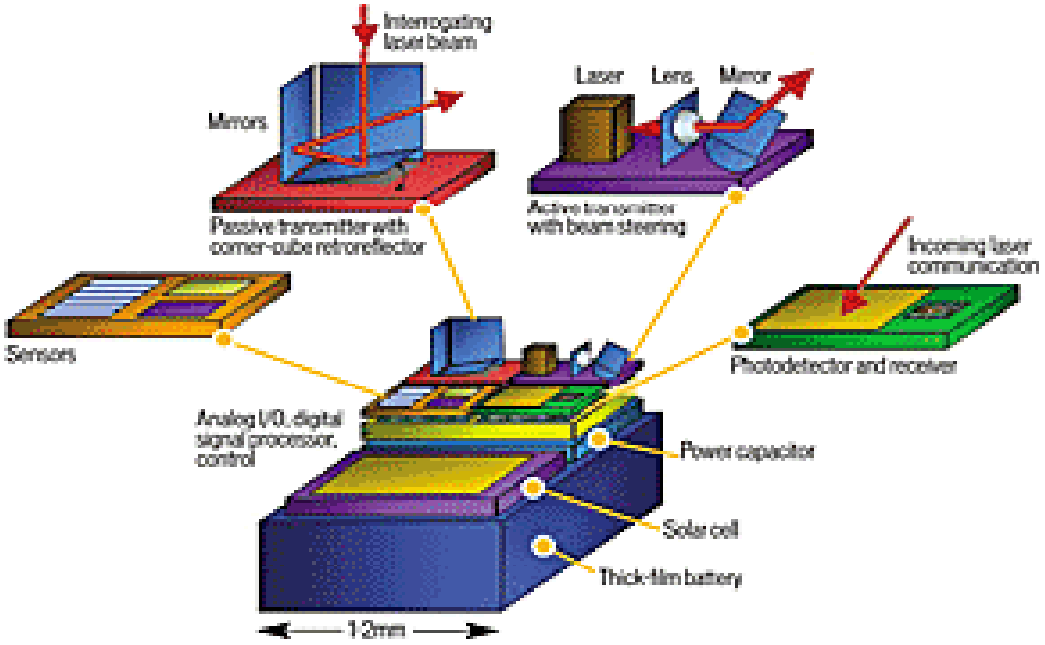






Multifunctional Mote

The goal of the smart-dust project at UC Berkeley is to build self-contained, millimeter-scale devices that include sensors, computational ability, bidirectional wireless communications technology and a power supply, while being inexpensive enough to be deployed by the hundreds.



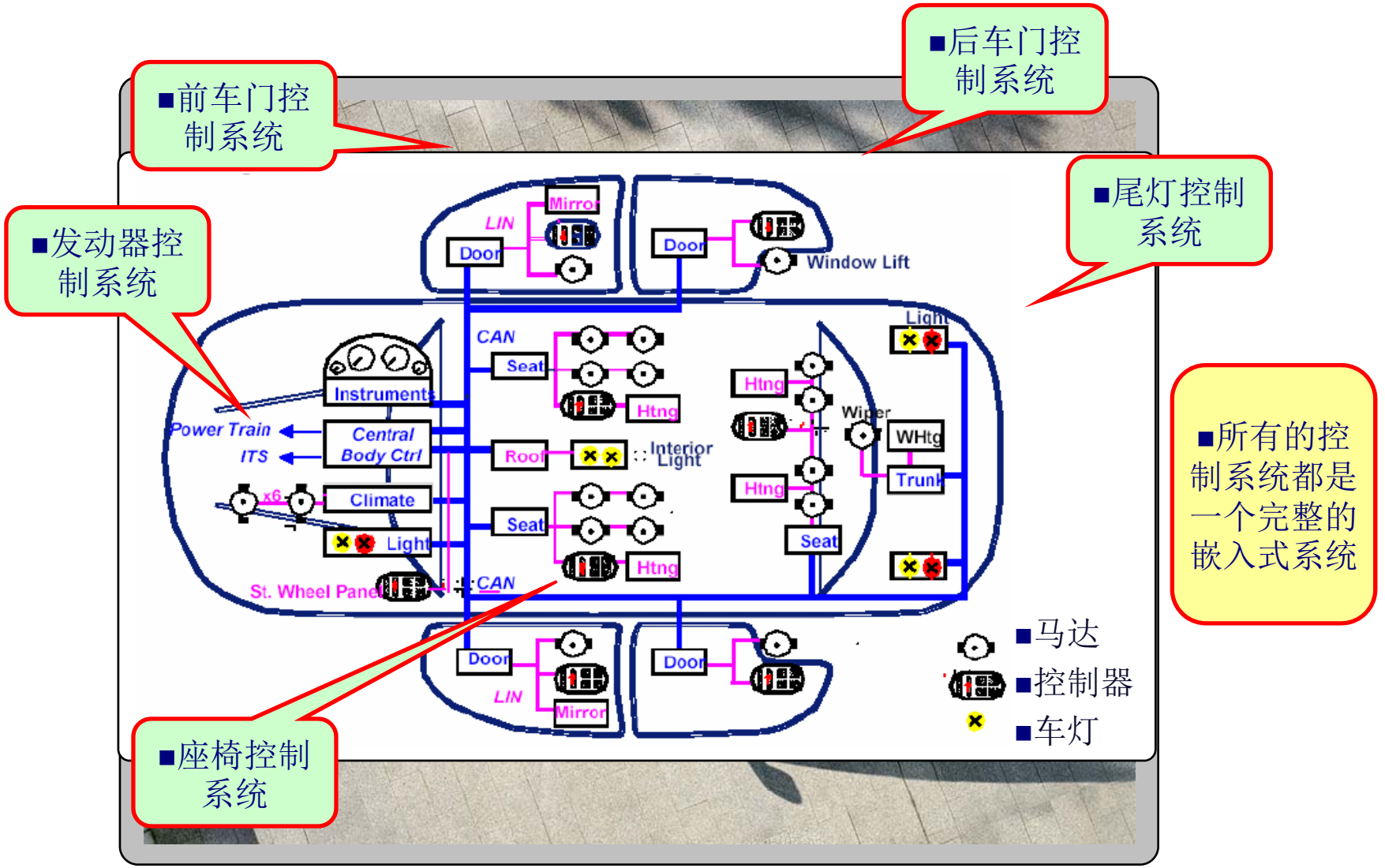
- http://robotics.eecs.berkeley.edu/~pister/SmartDust/

■Wearable Computing

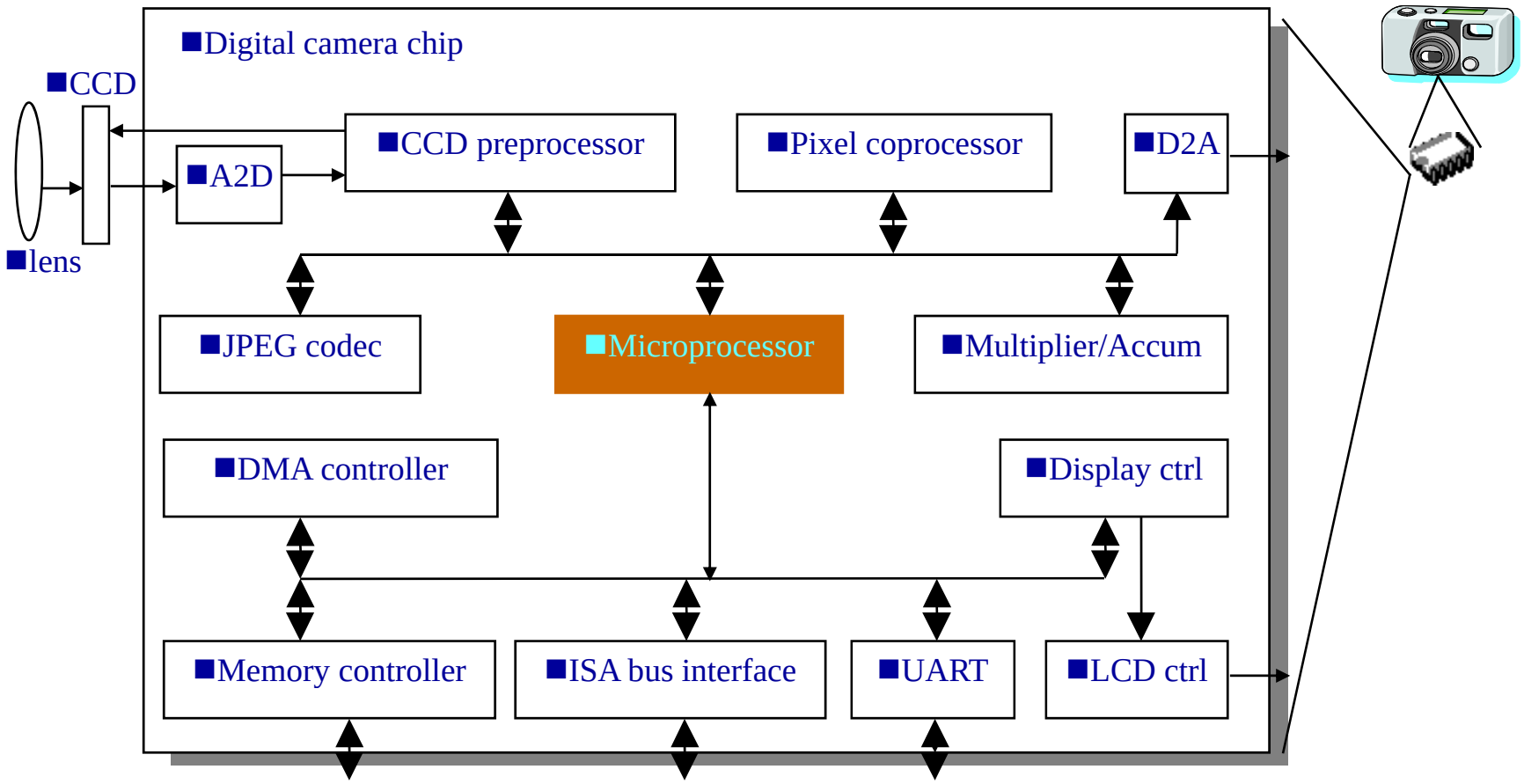
◆用于水下勘测的螃蟹机器人

英国巴斯大学一位研究生设计的，它目前可以在陆地上任意移动，未来的开发设计将计划将它用于完全的水下勘测任务操作。

■ 嵌入式系统示例——汽车控制系统



嵌入式系统的应用领域



A Digital Camera

◆ 嵌入式系统示例—Finepix数码相机的解剖照片

嵌入式工控机

- 工控机（Industrial Personal Computer，IPC）即工业控制计算机，是一种采用总线结构，对生产过程及机电设备、工艺装备进行检测与控制的工具总称。工控机具有重要的计算机属性和特征，如具有计算机CPU、硬盘、内存、外设及接口，并有操作系统、控制网络和协议、计算能力、友好的人机界面。
- 工控机的主要类别有：IPC（PC总线工业电脑）、PLC（可编程控制系统）、DCS（分散型控制系统）、FCS（现场总线系统）及CNC（数控系统）五种。

嵌入式工控机的优势

- 1、嵌入式工业触控平板电脑是一体机的结构，主机、液晶显示器、触摸屏合为一体，稳定性比较好。
- 2、采用目前比较流行的触摸功能，可以简化工作，更方便快捷，比较人性化。
- 3、嵌入式工业触控平板电脑体积较小，安装维护非常简便。
- 4、嵌入式工业触控平板电脑前面板大多采用铝镁合金压铸成型，坚固结实，持久耐用，而且重量比较轻。
- 5、大多数嵌入式工业触控平板电脑采用无风扇设计，利用大面积鳍状铝块散热，功耗更小，噪音也小。

嵌入式车载系统

- 车载终端是智能交通系统中关键的一环，车辆的所有信息都必须通过车载终端得到，而且车载终端还必须具备一定的处理能力，以便能够对搜集到的信息进行必要的处理，最后车载终端要能够接收并处理监控中心发出的命令。

■车载系统的嵌入式开发板

- 凯迪拉克CUE采用Linux操作系统和ARM 11三核处理器，可以实现400MIPS运算处理能力，比现有系统快3.5倍；由于采用开放的软件平台，这意味着全球软件开发者可以为搭载该系统的凯迪拉克车型开发应用程序，供车主下载使用。

2014苹果公司发布的车载系统CarPlay（映射）

其中一大亮点就是支持链接智能车载系统CarPlay，只要将用户的iPhone连接到启用了CarPlay的汽车，可支持“电话”、“音乐”、“地图”、“信息”和第三方音频应用程序，并可通过Siri，汽车触摸屏进行控制，为carplay提供了操作系统的支持。

发展趋势

- 信息时代，数字时代使得嵌入式产品获得了巨大的发展契机，为嵌入式市场展现了美好的前景，同时也对嵌入式生产厂商提出了新的挑战，从中我们可以看出未来嵌入式系统的几大发展趋势：
- 1．嵌入式开发是一项系统工程，因此要求嵌入式系统厂商不仅要提供嵌入式软硬件系统本身，同时还需要提供强大的硬件开发工具和软件包支持
- 2．网络化、信息化的要求随着因特网技术的成熟、带宽的提高日益提高，使得以往单一功能的设备如电话、手机、冰箱、微波炉等功能不再单一，结构更加复杂
- 3．网络互联成为必然趋势
- 4．精简系统内核、算法，降低功耗和软硬件成本
- 5．提供友好的多媒体人机界面

嵌入式系统应用走向纵深

- 进入20世纪90年代，在分布控制、柔性制造、数字化通信和数字化家电等巨大需求的牵引下，嵌入式系统的硬件、软件技术进一步加速发展、应用领域进一步扩大。
  - 手机、数码相机、VCD、数字电视、路由器、交换机等都是嵌入式系统。
  - 大多数豪华轿车每辆拥有约50个嵌入式微处理器。
  - 最新的波音777宽体客机上约有1000个微处理器。
  - 在不久的将来你会在你的家里发现几十到上百的嵌入系统在为你服务。

嵌入式系统应用走向纵深

- 嵌入式系统的硬件
  - 4位、8位、16位微处理器芯片已逐步让位于32位嵌入式微处理器芯片。
  - 面向不同应用领域的（Application-Specific）、功能强大、集成度高、种类繁多、价格低廉、低功耗的32位芯片已大量应用于各种各样的军用和民用设备。
  - DSP向高速、高精度、低功耗发展。
  - DSP与通用嵌入式微处理器集成（SoC）已成为现实，并已大量应用于嵌入式系统，如手机、IP电话等。

嵌入式系统应用走向纵深

- 在工业控制领域，嵌入式PC大量应用于嵌入式系统中。
- PC104、CPCI（Compact PCI）总线因其成本低、兼容性化也已被广泛应用。

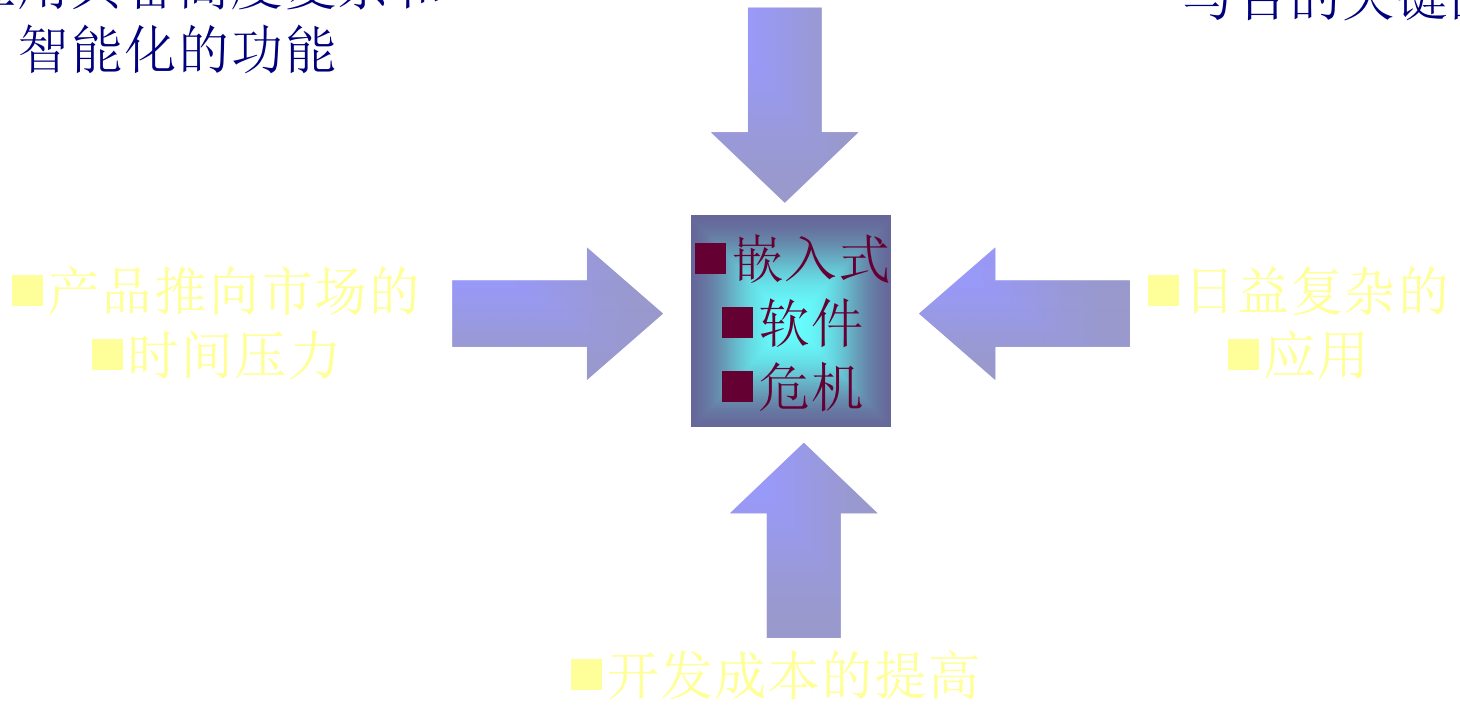
嵌入式系统的软件

- 随着微处理器性能的提高，嵌入式软件的规模也随着发生指数型增长。

嵌入式系统应用走向纵深

■32位芯片将能够执行由上百万行C代码构成的复杂程序，使得嵌入式应用具备高度复杂和智能化的功能

■软件的实现从某种意义上说决定了产品的功能，已成为新产品成功与否的关键因素。



嵌入式系统的发展趋势

- 嵌入式系统联网成为必然趋势，驱动了大量新的应用
  - 针对外部联网要求，嵌入系统必需配有通信接口，需要TCP/IP协议簇软件支持。
  - 针对内部联网要求，新一代嵌入式系统还需具备IEEE1394、USB、CAN、Bluetooth或IrDA通信接口，同时也需要提供相应的通信组网协议软件和物理层驱动软件。
  - 为了支持网络交互的应用，还需内置XML浏览器和Web Server。

第十讲　Cortex-M3架构和指令系统

§ 1嵌入式系统的应用领域

§ 2 ARM公司简介

§3　Cortex-M3内核

§4　指令系统

§5　嵌入式编程

§10.2 ARM公司概况

- 成立于1990年，总部在英国剑桥，目前拥有员工2000多名，分布在全球的32个分支机构。全球有700多家ARM Connected Community成员企业一起支持ARM的技术，成为业界最大的一个联盟。
- ARM自2001年起在中国市场发展，大力支持中国企业、科研机构、大学教育的创新。国内已有400多所大学开设了ARM相关的课程和实验室、出版了 120多本中文的ARM相关教科书。

§10.2 ARM公司概况

- ARM中国在上海、北京及深圳设有办事处,和国内70多家ARM Connected Community 成员企业一起支持ARM技术推广
- ARM 技术已在 95% 的智能手机、80% 的数码相机以及 35% 的所有电子设备中得到应用
- 向 250 多家公司出售了 800 个处理器许可证
- 目前为止已销售超过 200 亿个基于 ARM 的芯片

§10.2.1 ARM公司概况

- ARM中国在上海、北京及深圳设有办事处,和国内70多家ARM Connected Community 成员企业一起支持ARM技术推广。
- ARM 成为MCU 制作方法(IP)最大的提供商；
- 2016年7月，软银宣布以320亿美元收购英国芯片设计公司ARM，好像全世界都不太开心。

§10.2.2 ARM公司的主要产品

IP Products

Processors
Cortex-A
Cortex-R
Cortex-M
Classic Processors
Machine Learning

Physical IP
Logic IP
Memory Compilers
Interface IP
POP IP

System IP
CoreLink Interconnect
CoreLink System Controllers
CoreSight Debug and Trace
Memory Controllers
TrustZone Security IP
Free System IP Whitepapers

Graphics and Multimedia Processors
Mali GPUs
Mali Video Processors
Mali Display Processors
Mali Camera
Assertive Display

System Design Tools
System Design Kits
System Guidance
Subsystems
Socrates System Builder
Fast Models
Cycle Models
Development Boards
Fixed Virtual Platforms

Software Tools
Graphics Development Tools
Solutions
Product Enquiries
Compilers
HPC
DS-5 Development Studio
Keil MDK
Debug Probes and Adapters

§10.2.2 ARM公司的主要产品

2017年3月，推出全新芯片架构DynamIQ，通过这项技术，AI的性能可提升50倍。

1）在V8.3指令集中增加面向人工智能的专用指令；

2）在单个集群（Cluster）中支持大小核或异构核；

3）实现每个核独立的电压频率调节；

4）可用于汽车无人驾驶系统。

§10.2.2 ARM公司的主要产品

- AI的3大部件

§10.2.2 ARM公司的主要产品

2016年

- Cortex-R8：4个32位实时内核，针对5G手机市场
- Cortex-A32：ARMv8-A 架构，嵌入式与物联网
- Cortex-A73：64位，多核，性能更强的移动SoC
- Mali-DP650：4K移动显示处理器
- Cortex-M33：基于ARMv8-M架构的嵌入式处理器

具备功能配置选项，包括协处理器接口、DSP和浮点计算，相较Cortex-M3 和 Cortex-M4拥有更出色的性能与能效表现。

§10.2.2 ARM公司的主要产品

2015年

- Cortex-R5处理器，安全文件集加速，进军汽车、医疗与工业应用市场
- ARM在中国发布“全球大学计划合作联盟”项目

2014年

- ARM Cortex-M7
- 免费mbed OS，ARM mbed™ 物联网设备平台
- Mali-T800图形处理器

2013年

- ARM Mali-T760图形处理器

§10.2.2 ARM公司的主要产品

2012年

- 10月推出64位 Cortex-A50系列
- ARM发布Cortex-A15 四核处理器硬宏
- ARM推出全球最节能® Cortex™-M0+处理器，锁定低价微控制器、传感器和控制器市场 ，耗电量仅9uA/MHz；

2011年

- 公开下一代ARM架构技术细节
- MALI-T658 GPU 进一步拓展在高性能产品图形处理和GPU 计算领域的领导地位
- ARM 通过 Keil MDK（专业版）为 Cortex-M 系列处理器扩展了软件开发工具

§10.2.2 ARM公司的主要产品

2010年

- ARM 为实现高性能的数字信号控制推出了 Cortex-M4 处理器
- ARM Mali 成为被最广泛授权的嵌入式 GPU 架构

2009年

- ARM 宣布实现 2GHz 频率的 Cortex-A9 双核处理器
- ARM 推出体积最小、功耗最低和能效最高的处理器 Cortex-M0

§10.2.2 ARM公司的主要产品

2008年

- ARM 宣布销售 100 亿台处理器
- ARM Mali-200 GPU 全球第一个实现在 1080p HDTV 分辨率下符合 Khronos Open GL ES 2.0 标准的产品；

2007年

- 向移动设备市场售出 50 亿台 ARM Powered 处理器
- 发布了 ARM Cortex-M1 处理器，它是第一款专为 FPGA 中的实现设计的 ARM 处理器
- ARM 推出 Cortex-A9 处理器以实现可扩展性能和低功耗设计
- ARM 推出针对智能卡应用的 SecurCore SC300 处理器

§10.2.2 ARM公司的主要产品

2006年

- IEEE 为 ARM 颁发 2006 年度企业创新成就奖
- ARM Cortex-A8 处理器被 4 家著名的电子行业出版社评选为“2005 年最佳处理器

2005年

- ARM 收购了 Keil Software
- ARM 发布 Cortex-A8 处理器

2004年

- ARM 发布作为新型 Cortex 处理器内核系列中首款的 Cortex-M3 处理器
- 发布第一款集成多处理器，即 MPCore 多处理器

§10.2.2 ARM公司的主要产品

2003年

- ARM 宣布到目前为止已销售 10 亿多颗微处理器核
- ARM 发布 ARM11 微架构
- ARM 发布 RealView 开发工具系列

2001年

- ARM 在 32 位嵌入式 RISC 微处理器市场的份额已增至 76.8%
- ARM 发布新型 ARMv6 架构

2000年

- ARM 发布 SecurCore 智能卡系列
- ARM 发布可合成的 ARM9E 处理器，提高了信号的处理能力

§10.2.2 ARM公司的主要产品

1999年 发布可合成的 ARM9E 处理器，提高信号处理能力

1998年 开发了可合成的 ARM7TDMI 核心版本

1997年 发布 ARM9TDMI 系列

1996年 ARM 和 Microsoft 联袂将 Windows CE 扩展到 ARM 架构上

1995年 发布 Thumb 架构扩展；软件开发工具包

1993年 推出 ARM7 核

1991 年 推出第一款嵌入式 RISC 核，即 ARM6

§10.2.3 ARM公司的商业模式

ARM公司的商业模式是提供技术许可的知识产权，而不是制造和销售实际的半导体芯片。

ARM的合作伙伴包括世界领先的半导体和系统公司。全球目前有200多家（包括最大的前20家）半导体公司已从ARM公司获得技术授权，成为ARM的客户。

模式好: 避开与INTEL等直接竞争，化敌为友。

时机好: 半导体厂商推MCU需要设计自己CPU核，开发成本、推广成本很高。

§10.2.4 ARM微处理器的特点

目前市场上主要的32位微处理器有PowerPC、MIPS、ARM 这三大类型。ARM处理器为RISC芯片，其简单的结构使ARM内核非常小，使得器件的功耗也非常低。具有经典RISC的特点：

1）大的、统一的寄存器文件

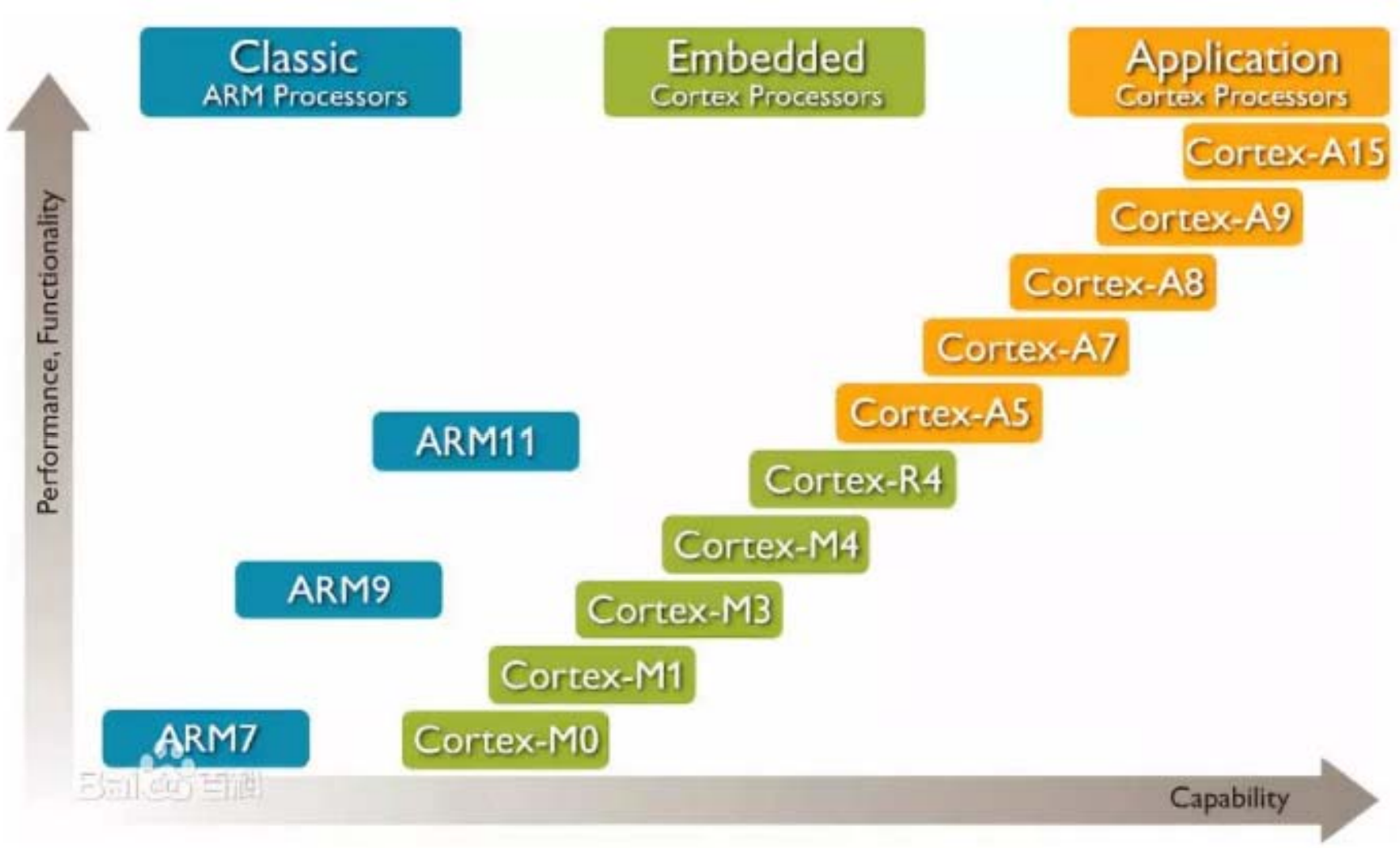
2）装载/保存结构，数据处理 操作只针对寄存器的内容，而不直接对存储器进行操作；

3）简单的寻址模式;

4）统一和固定长度的指令域，简化了指令的译码。

RISC的英文全称是Reduced Instruction Set Computer，中文是精简指令集计算机。特点是所有指令的格式都是一致的，所有指令的指令周期也是相同的，并且采用流水线技术。在中高档服务器中采用RISC指令的CPU主要有Compaq（康柏，即新惠普）公司的Alpha、HP公司的PA-RISC、IBM公司的PowerPC、MIPS公司的MIPS和SUN公司的Sparc。

§10.2.5 ARM的常见种类



§10.2.5 ARM的常见种类

1）经典处理器：ARM系列

ARM7：70MHz，成本低，应用最广；

ARM9：200-500MHz，运算能力较强、应用多

ARM11：ARM9的性能提升，应用较少

推出时间已超过 15 年， ARM7TDMI®仍是市场上销量最高的 32 位处理器。每个季度的设备销量超过 10 亿台，或者每秒超过 90 台，并且颁发的许可证超过 500 个。此类范围广泛的处理器占据目前市场上销售的所有电子产品的四分之一，仍处于核心地位

§10.2.5 ARM的常见种类

1）经典处理器：ARM系列

经典 ARM 处理器	许可证数
ARM11 系列	82
ARM9 系列	271
ARM7 系列	172

§10.2.5 ARM的常见种类

Accent
Altera Corporation
AMI Semiconductor
Analog Devices Inc.
Atmel Corporation
Avalink Incorporated
Beken Corporation
Broadcom Corporation
GLOBALFOUNDRIES
Chengdu Javee Microelectronics Co.
Cirrus Logic
Conexant Systems Inc.
Dialog Semiconductor
DSPG
eSilicon Corporation
Freescale Semiconductor
Fujitsu
Gainspan Corporation
Global Unichip Corporation
Hong Kong Science and Technology Parks
Indilinx
Infineon Technologies AG
Intel Corporation
Intellon Corporation
Intrinsic Corporation
Kawasaki Microelectronics
LSI Logic
Mamurian Design Inc
Mediatek
Micronas
NEC Electronics
Neo Magic Corporation
Nuvoton Technology Corporation
NVIDIA
NXP (LPC2000系列)
OKI
Panasonic
Pixim
PLX Technology Inc
Qualcomm
RDA International Inc
Rohm
Samsung Electronics
Sanyo
Seiko Epson
Shanghai Fudan
Sharp
Shenzhen State Microelectronics (SMIT)
SiRF Technology
Skyworks
Socle Technology Corp
Sony
Spreadtrum Communications Inc.
STMicroelectronics
Telegent Systems Inc
Texas Instruments
Toshiba
U-Blox AG
Verisilicon
Via Telecom/Via Technology
Xi'an Huaxun
Yamaha Corporation
Zarlink Semiconductor
Zoran Corporation

2）Cortex-A系列：高性能

2H 2014, Expiration Q1 2015

ARM®

§10.2.5 ARM的常见种类

2）Cortex-A系列：高性能

- Cortex-A15：移动应用，性能最高的解决方案
- Cortex-A7：800 MHz - 1.2 GHz 的典型频率
- Cortex-A9：800 MHz - 2 GHz 的标准频率，每个内核可提供 5000 DMIPS 的性能
- Cortex-A8： 单核解决方案，可提供经济有效的高性能，在 600 MHz - 1 GHz 的频率下，性能超过 2000 DMIPS
- Cortex-A5：低成本实现，在 400- 800 MHz 的频率下，提供的性能超过 1200 DMIPS。

上述处理器都支持 ARM 的第二代多核技术

§10.2.5 ARM的常见种类

3）Cortex-R系列：实时处理器

Cortex-R4、Cortex-R5 和 Cortex-R7 处理器用于嵌入式实时产品（如汽车安全或无线基带）等

- 硬件除法器、浮点单元 (FPU) 选项
- 具有 Thumb-2 指令的 ARM v7-R 架构，高代码密度
- 指令集增强，包括 SIMD、 DSP 和媒体处理
- 具有指令和数据高速缓存控制器的哈佛架构
- 用于获得快速响应代码和数据的处理器本地的紧密耦合内存 (TCM)
- 高性能 64 位 AMBA 3 AXI 总线接口
- 奇偶校验检测和 ECC，用于 1 级内存系统和总线的软错误和硬错误检测/更正等。

§10.2.5 ARM的常见种类

3）Cortex-M系列：全球微控制器标准

- Cortex-M4: 168MHz，高性能，硬件浮点
- Cortex-M3: 普及型，高性价比，取代ARM7
- Cortex-M0: 高性价比，取代8、16位mcu
- Cortex-M0+: 超低功耗，取代微功耗8、16位mcu
- 已许可给 40 个以上的 ARM 合作伙伴，包括 NXP 、STMicroelectronics(M3,性价比)、Freescale(M4,模拟特性)、Texas Instruments 等领先供应商。

第十讲　Cortex-M3架构和指令系统

§1嵌入式系统的应用领域

§2 ARM公司简介

§3　Cortex-M3内核

§4　指令系统

§5　嵌入式编程

§10.3 Cortex-M3内核简介

一、CM3的内核组成

二、三级流水线

三、 Thumb2指令集

四、工作模式和访问特权

五、内部寄存器

六、中断机制

七、复位功能

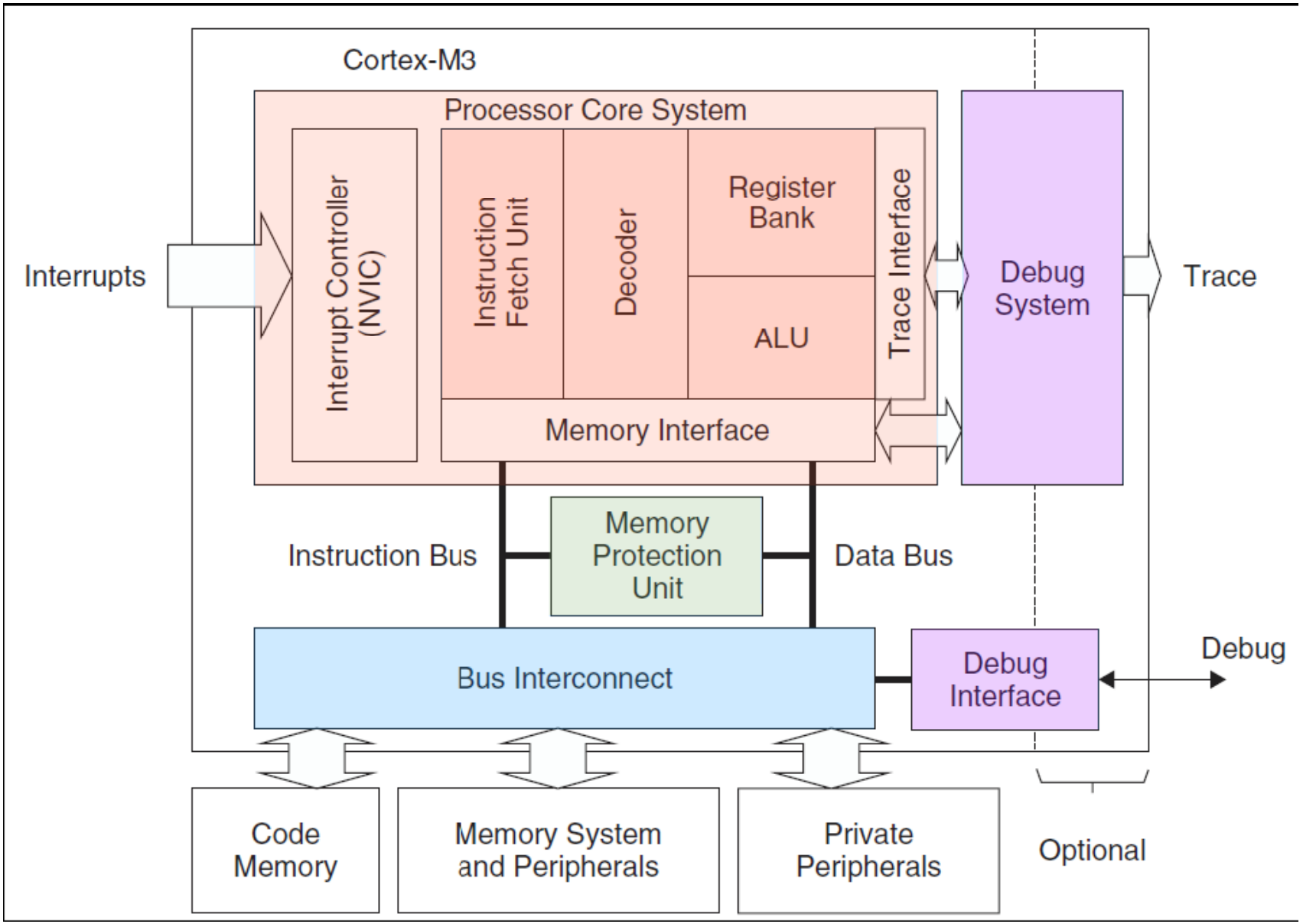
八、存储器及其映射

九、调试接口

§10.3 Cortex-M3内核简介

§10.3.1 Cortex-M3组成（ARM7的简化、改进版）

比较项目	ARM7	Cortex-M3
架构	ARMv4T（冯诺依曼） 指令和数据总线共用，会出现瓶颈	ARMv7-M（哈佛） 指令和数据总线分开，无瓶颈
指令集	32位ARM指令+16位Thumb指令 两套指令之间需要进行状态切换	Thumb/Thumb-2指令集 16位和32位 指令可直接混写，无需状态切换
流水线	3级流水线 若出现转移则需要刷新流水线，损失惨重	3级流水线+分支预测 出现转移时流水线无需刷新，几乎无损失
性能	0.95DMIPS/MHz（ARM模式）	1.25DMIPS/MHz
功耗	0.28mW/MHz	0.19mW/MHz
低功耗模式	无	内置睡眠模式
面积	0.62mm2（仅内核）	0.86mm2（内核+外设）
中断	普通中断IRQ和快速中断FIQ太少，大量外设不得不复用中断	不可屏蔽中断NMI+1-240个物理中断 每个外设都可以独占一个中断，效率高
中断延迟	24-42个时钟周期，缓慢	12个时钟周期，最快只需6个
中断压栈	软件手工压栈，代码长且效率低	硬件自动压栈，无需代码且效率高
存储器保护	无	8段存储器保护单元（MPU）
内核寄存器	寄存器分为多组、结构复杂、占核面积多	寄存器不分组（SP除外），结构简单
工作模式	7种工作模式，比较复杂	只有线程模式和处理模式两种，简单
乘除法指令	多周期乘法指令，无除法指令	单周期乘法指令，2-12周期除法指令
位操作	无 访问外设寄存器需分“读-改-写”3步走	先进的Bit-band位操作技术，可直接访问外设寄存器的某个值
系统节拍定时	无	内置系统节拍定时器，有利于操作系统移植



Cortex - M3 的一个简化视图

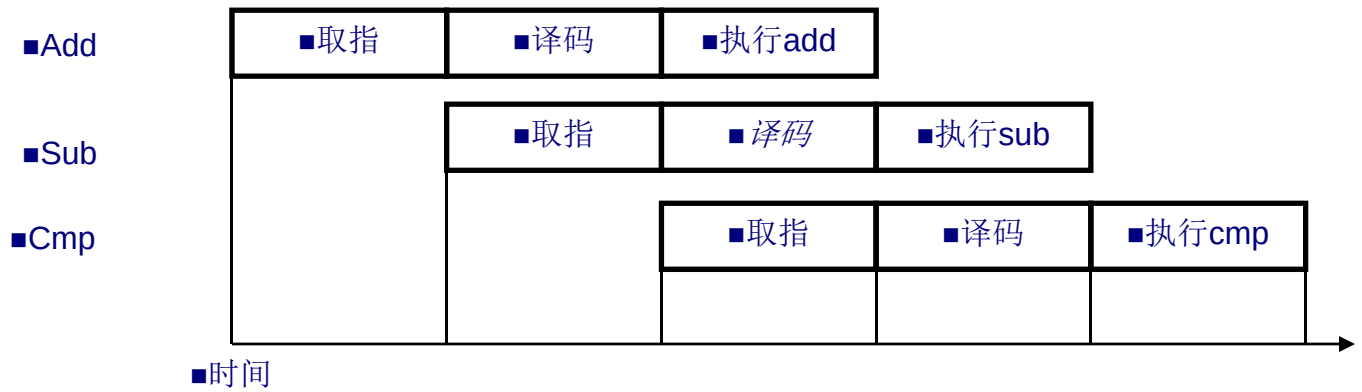
来源： STM32权威指南.pdf

§10.3.2 三级流水线

CM3处理器使用流水线+分支预测的操作模式，使几个操作同时进行，并使处理和存储器系统连续操作，当出现转移时流水线无需刷新，几乎无损失，增加了处理器指令流的速度。

CM3的流水线分3级，分别为：

取指→译码→执行



§10.3.2 三级流水线

正常操作过程中，在执行一条指令的同时对下一条指令进行译码，并将第三条指令从存储器中取出。这三条指令之间的位置关系如下表所示：

流水线上各指令的地址		流水线工位	描述
ARM指令集	Thumb指令集		
PC	PC	取指	指令从存储器中取出
PC-4	PC-2	译码	对指令使用的寄存器进行译码
PC-8	PC-4	执行	从寄存器组中读出寄存器，执行移位和ALU操作，寄存器被写回到寄存器组中

§10.3.3 一个指令集

Cortex-M3 把ARM7处理器的有32位ARM和16位Thumb两种指令集，改进为16位或32位的Thumb-2 以及16位Thumb指令集，但不能执行ARM指令。

避免了ARM7在两种状态切换过程可能出现的错误，简化了应用程序开发。

§10.3.4 两种工作模式

1）线程模式和处理模式

目的：引入处理模式（handle mode）和线程模式(thead mode) 的本意，是用于区别普通应用程序（线程模式）和中断服务程序（处理模式）；

效果：有效的简化了 ARM7体系结构支持7种处理器模式(用户模式、快中断模式、中断模式、管理模式、中止模式、未定义模式和系统模式)。

§10.3.4 两种工作模式

2）特权级和用户级

目的：用于存储器访问的保护机制。使得普通的用户程序代码不能意外地，甚至是恶意地执行涉及到要害的操作；

特权级：该级别的程序可以访问所有范围的存储器（如果有MPU，还要在MPU 规定的禁地之外），并且可以执行所有指令；

用户级：用户级程序不能直接改写CONTROL 寄存器，需执行一条系统调用指令(SVC)，由异常服务例程修改CONTROL 寄存器，才能在用户级的线程模式下重新进入特权级。

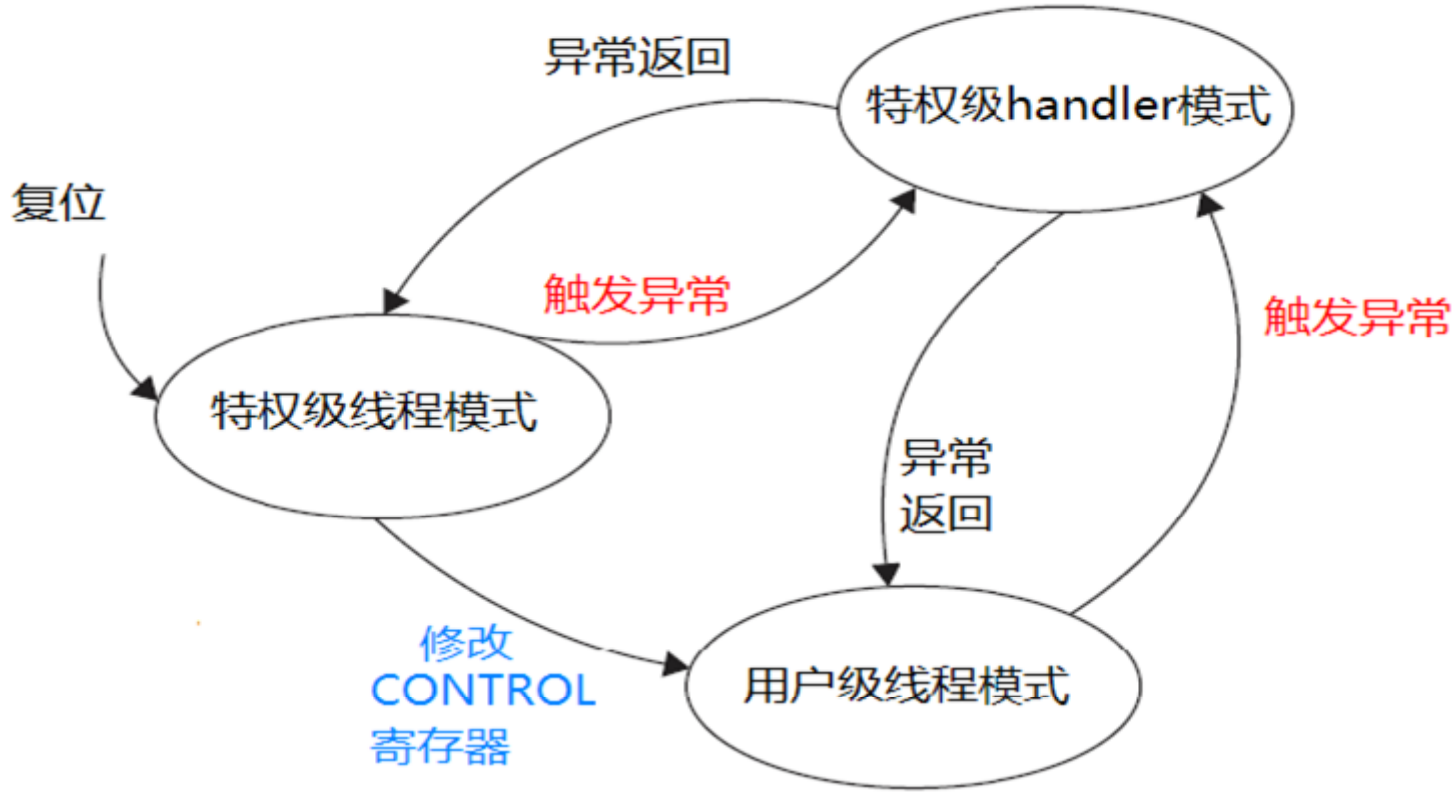
§10.3.4 两种工作模式

3）默认状态

- 在CM3 运行主程序（后台程序）时是线程模式，既可以使用特权级，也可以使用用户级；
  - 中断(异常)服务程序必须在特权级下执行；
- 复位后，mcu默认进入线程模式，特权极访问。

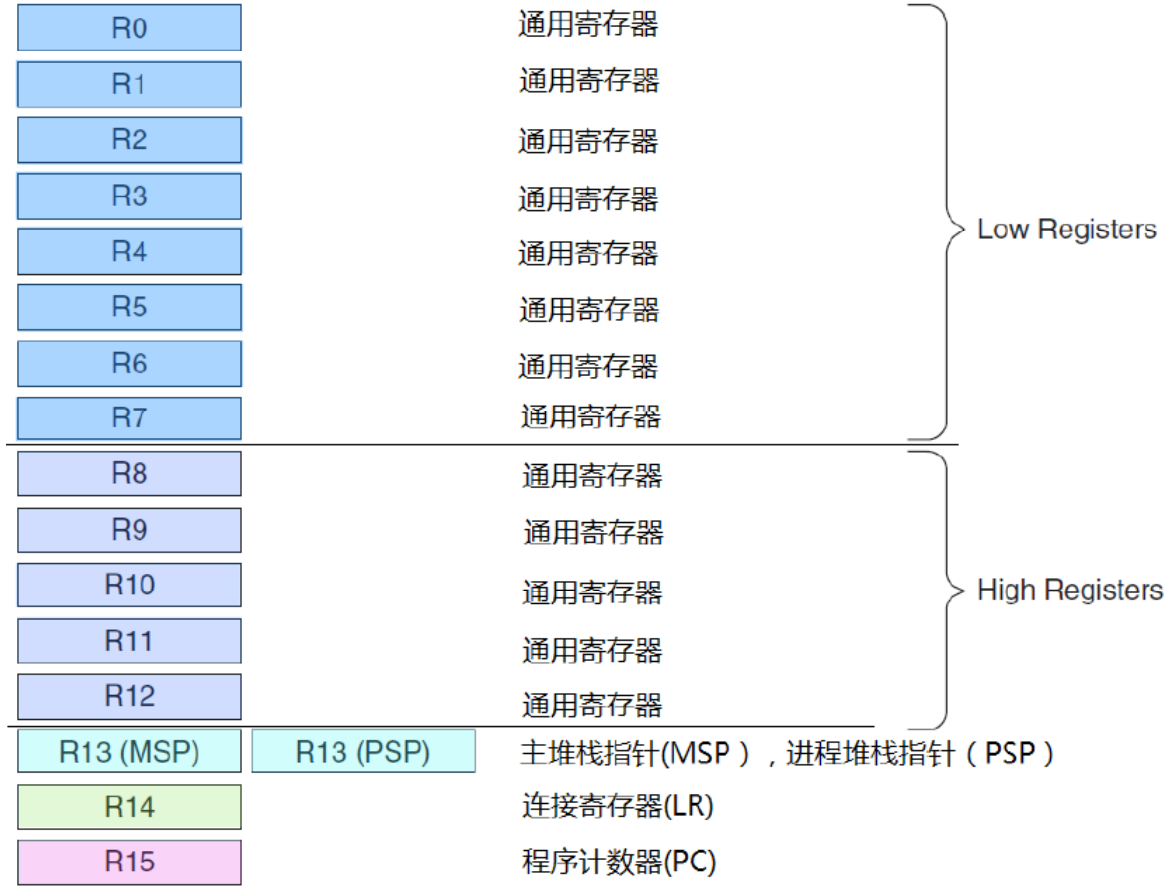
§10.3.4 两种工作模式

4）模式转换



§10.3.5 内部寄存器

1）组成：CM3 处理器拥有R0 - R15 的寄存器组。其中R13 作为堆栈指针SP，SP 有两个，但在同一时刻只能有一个可以使用。



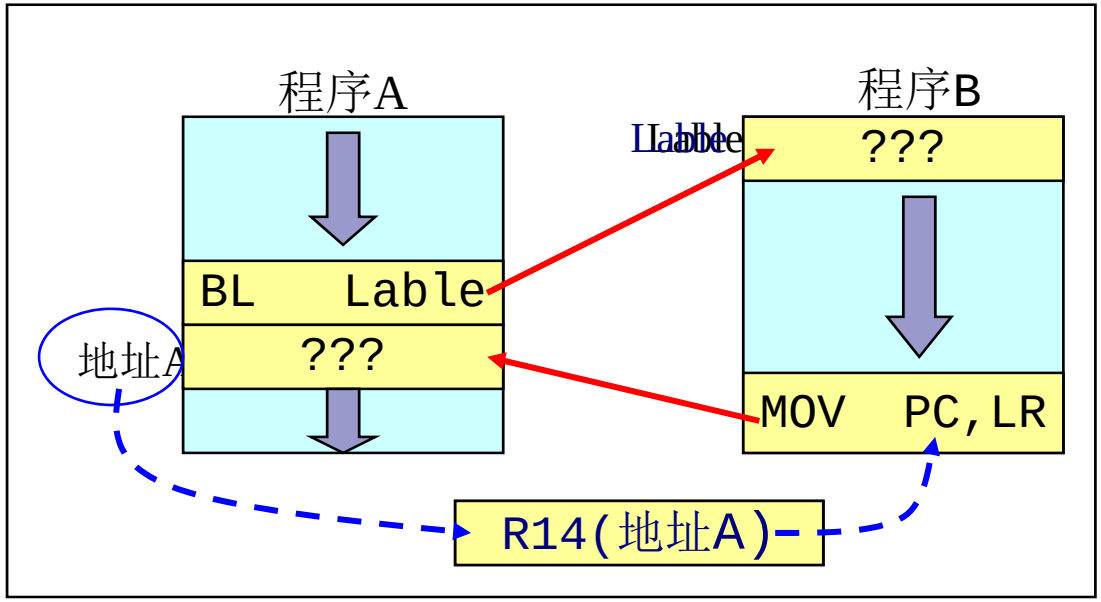
§10.3.5 内部寄存器

2）R14连接寄存器：用于存放子程序的返回地址调用

S1.程序A执行过程中调用程序B；

S2.程序跳转至标号Lable，执行程序B。同时硬件将“BL Lable”指令的下一条指令所在地址存入R14；

S3.程序B执行最后，将R14寄存器的内容放入PC，返回程序A；



§10.3.5 内部寄存器

3）R15程序指针：读R15的限制

正常操作时，从R15读取的值是处理器正在取指的地址，即当前正在执行指令的地址加上8个字节（两条ARM指令的长度）。由于ARM指令总是以字为单位，所以R15寄存器的最低两位总是为0。

地址	程序代码	流水线状态
PC-8	LDR R0,PC	正在执行
PC-4	???	正在译码
PC	???	正在取指

§10.3.5 内部寄存器

3）R15 程序指针：读R15的限制

当使用STR或STM指令保存R15时，会有一个例外。这些指令可能将当前指令地址加8字节或加12字节保存（将来可能还有其它数字）。偏移量是8还是12取决于具体的ARM芯片，但是对于一个确定的芯片，这个值是一个常量。

所以最好避免使用STR和STM指令来保存R15，如果很难做到，那么应当在程序中计算出该芯片的偏移量。

§10.3.5 内部寄存器

4）写R15的限制

正常操作时，写入R15 的值被当作一个指令地址，程序从这个地址处继续执行（相当于执行一次无条件跳转）。

由于ARM指令以字节为边界，因此写入R15的值最低两位通常为0b00。写入R15的值的最低两位为0，如果不是，结果将不可预测.

§10.3.6 中断机制

1）重要性

嵌入式系统的实时性通过MCU的中断功能来时实现，因此中断机制和中断部件使用非常重要

2）中断流程

当预定义的事件发生时，正常程序被暂停，处理器进入异常模式。自动将处理器状态保存到堆栈中，执行中断服务程序（ISR），结束时自动从堆栈中恢复，继续执行被暂停的正常程序

3）高效率

嵌套向量中断控制器（NVIC）支持末尾连锁（tail-chaining）中断技术，执行背对背中断（back-to-back interrupt），能够省略连续中断之间的状态保存和恢复指令

§10.3.6 中断机制

4）其它特点

- 中断优先级可动态重新设置
- 中断数目可配置为1～240
- 中断优先级的数目可配置为1～8 位（1～256 级）
- 处理模式和线程模式具有独立的堆栈和特权等级
- 使用C/C++标准的调用规范：*ARM 架构的过程调用标准（PCSAA）执行ISR 控制*传输。

异常入口/出口汇总

异常类型	位置	优先级	描述
-	0	-	在复位时栈顶从向量表的第一个入口加载。
复位	1	-3（最高）	在上电和热复位（warm reset）时调用。在第一条指令上优先级下降到最低（线程模式）。异步的。
不可屏蔽的中断	2	-2	不能被除复位之外的任何异常停止或占先。异步的
硬故障	3	-1	由于优先级的原因或可配置的故障处理被禁止而导致不能将故障激活时的所有类型故障。同步的
存储器管理	4	可调整 [a]	MPU 不匹配，包括违反访问规范以及不匹配。是同步的。即使 MPU 被禁止或不存在，也可以用它来支持默认的存储器映射的 XN 区域。
总线故障	5	可调整 [b]	预取故障，存储器访问故障，以及其它相关的地址/存储故障。精确时是同步，不精确时是异步。
使用故障	6	可调整	使用故障。例如，执行未定义的指令或尝试不合法的状态转换。是同步的。
-	7-10	-	保留
SVCall	11	可调整	利用 SVC 指令调用系统服务。是同步的。
调试监控	12	可调整	调试监控，在处理器没有停止时出现。是同步的，但只有在使能时是有效的。如果它的优先级比当前有效的异常的优先级要低，则不能被激活。
-	13	-	保留
PendSV	14	可调整	可挂起的系统服务请求。是异步的，只能由软件来实现挂起。
SysTick	15	可调整	系统节拍定时器（tick timer）已启动。是异步的。
外部中断	16 及以上	可调整	在内核的外部产生。**INTISR[239:0]**，通过 NVIC（设置优先级）输入，都是异步的。

a　该异常的优先级可修改，见*系统处理器优先级寄存器的位分配*。可调整的范围为 NVIC 的 0～N 优先

§10.3.6 中断机制

- 优先级

通过对中断优先级寄存器的8 位PRI_N 区执行写操作，将中断的优先级指定为0～255，（0 优先级最高，255 优先级最低）

软件优先级的设置对复位，NMI，和硬故障无效。它们的优先级始终比外部中断要高。

如果两个或更多的中断指定了相同的优先级，则由它们的硬件优先级来决定处理器对它们进行处理时的顺序。

§10.3.6 中断机制

- 2个独立的堆栈

进程堆栈（process stack），SP_process 为进程堆栈的SP 寄存器。线程模式在复位后使用主堆栈主堆栈，可以配置为使用进程堆栈。

主堆栈（main stack）， SP_main 为主堆栈的SP 寄存器。处理模式使用主堆栈，在将8 个寄存器压栈之后，ISR 使用主堆栈，并且后面所有的抢占中断都使用主堆栈。

§10.3.6 中断机制

- 压栈操作

当MCU进入中断时，它自动将下面的8个寄存器按以下顺序压栈：

PC

xPSR

r0~r3

r12

LR

- 出栈操作：顺序相反的操作

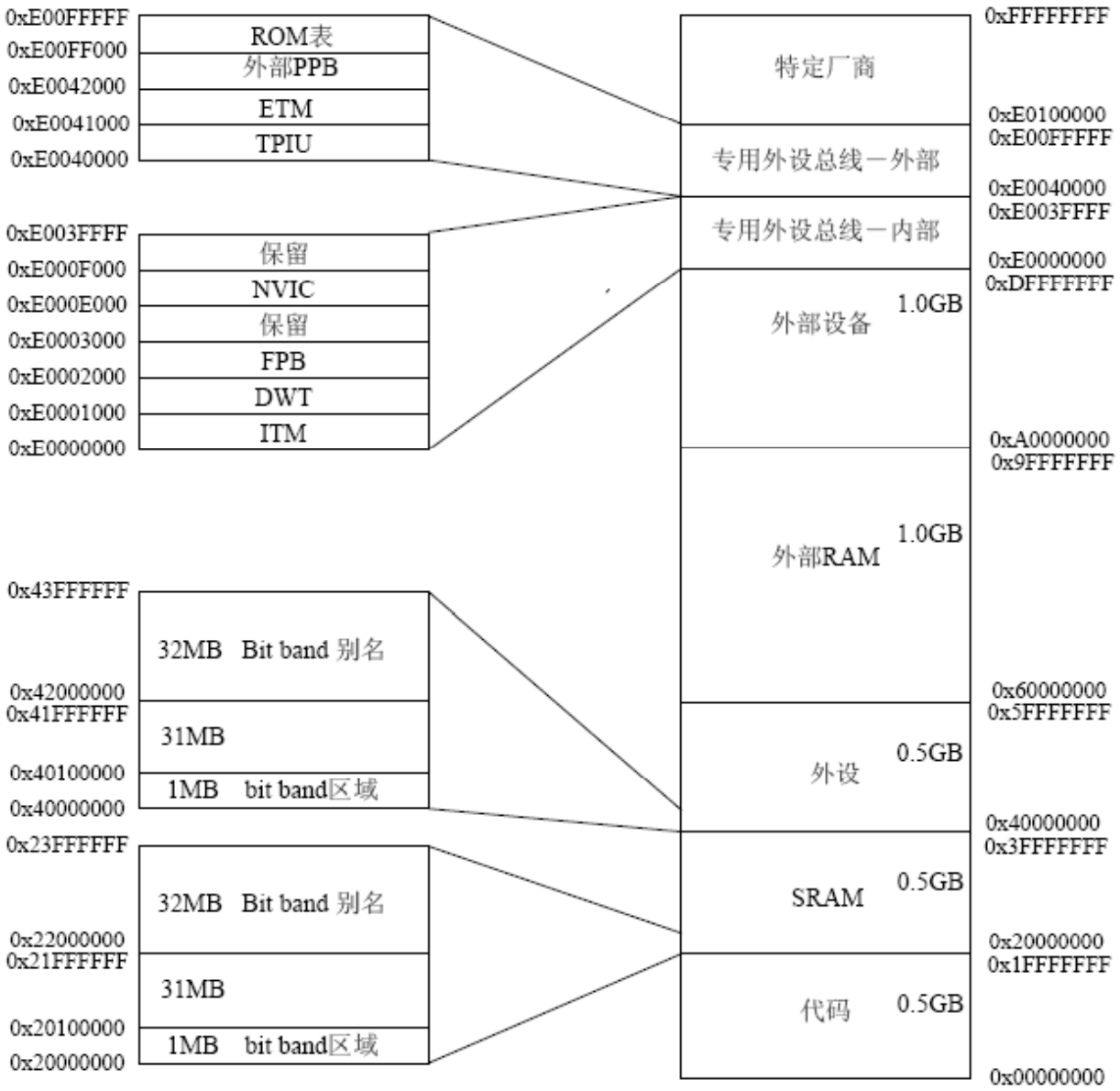
§10.3.7 复位功能

CM3 处理器含3 个复位输入：

复位输入	描述
PORESETn	复位整个处理器系统，JTAG-DP 除外
SYSRESETn	复位整个处理器系统，NVIC 中的调试逻辑、FPB、DWT、ITM 以及 AHB-AP 除外
nTRST	复位 JTAG-DP

复位方式	SYSRESETn	nTRST	PORESETn	应用
上电复位	x	0	0	接通电源后复位，复位整个系统。冷复位
系统复位	0	x	1	复位处理器内核和系统元件，调试元件除外
JTAG-DP 复位	1	0	1	复位 JTAG-DP 逻辑

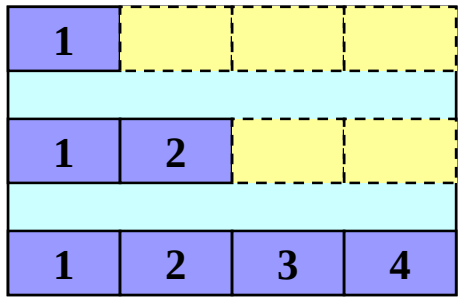
§10.3.8 存储器及存储器映射



§10.3.8 存储器及其映射

1）硬件直接支持的数据类型

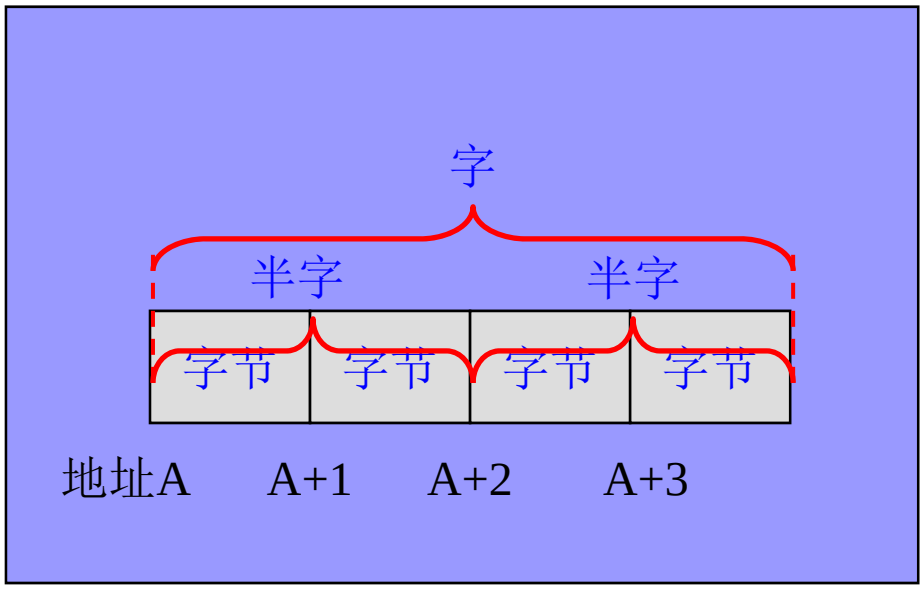
- 字节： 8位
- 半字：16位（必须分配为占用两个字节）
- 字：32位（必须分配为占用4各字节）



§10.3.8 存储器及其映射

2）存储器格式

- 地址空间的规则：
- 位于地址A的字包含的字节位于地址A,A+1,A+2和A+3；
- 位于地址A的半字包含的字节位于地址A和A+1；
- 位于地址A+2的半字包含的字节位于地址A+2和A+3；
- 位于地址A的字包含的半字位于地址A和A+2；



§10.3.8 存储器及其映射

3）小端模式和大端模式

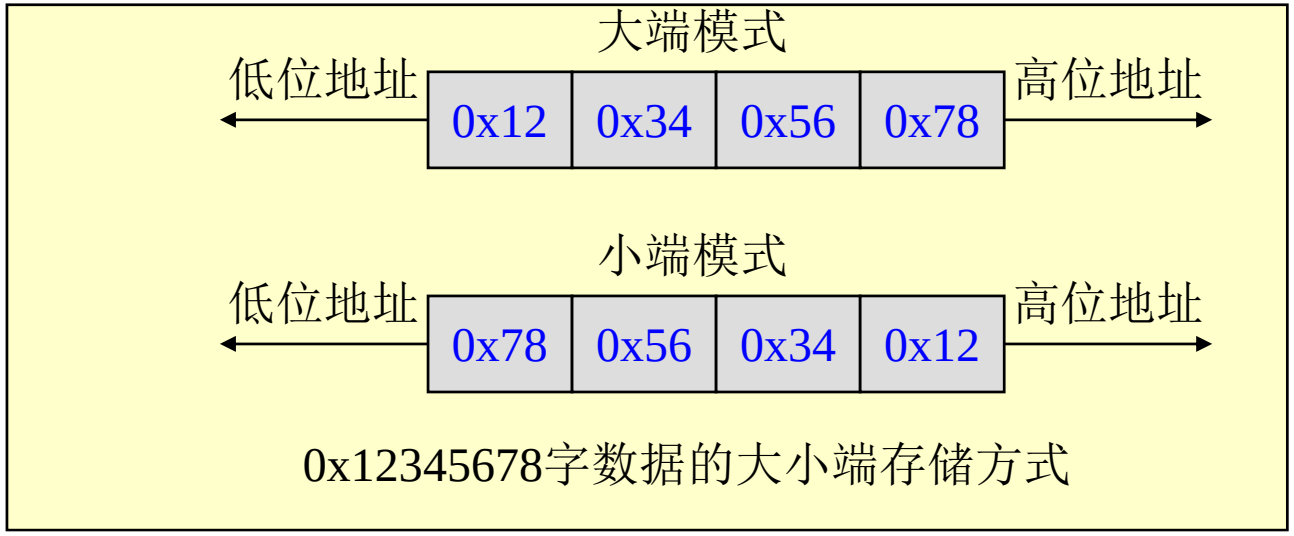
存储器系统有两种映射机制：

- 小端存储器系统：

在小端格式中，高位数字存放在高位字节中。因此存储器系统字节0连接到数据线7～0。

- 大端存储器系统：

在大端格式中，高位数字存放在低位字节中。因此存储器系统字节0连接到数据线31～24。



0x12345678字数据的大小端存储方式

§10.3.8 存储器及其映射

3）小端模式和大端模式

一个基于ARM内核的芯片可以只支持大端模式或小端模式，也可以两者都支持。

在ARM指令集中不包含任何直接选择大小端的指令，但是一个同时支持大小端模式的ARM芯片可以通过硬件配置（一般使用芯片的引脚来配置）来匹配存储器系统所使用的规则。

注意：如果实际的存储器格式与芯片的存储器格式不符时，只有以字为单位的数据存取才正确，否则将出现不可预期的结果。

§10.3.8 存储器及其映射

4）未对齐的存储器访问

ARM结构通常希望所有的存储器访问都合理的对齐。具体来说就是字访问的地址通常是字对齐的，而半字访问使用的地址是半字对齐的。不按这种方式对齐的存储器访问称为非对齐的存储器访问。

- 将一个非字（半字）对齐的地址写入ARM（Thumb）状态的R15寄存器，将引起非对齐的指令取指。
- 在一个非字（半字）对齐的地址读写一个字（半字），将引起非对齐的数据访问。

§10.3.9 调试接口

CM3处理器的调试接口有2种：

1) JTAG: 6线制接口

2) SWD: 2线制接口

§3 Cortex-M3内核简介

CM3内核资料参见：

- Cortex-M3 Technical Reference Manual.pdf (英文，ARM)
- Cortex-M3技术参考手册.pdf(中文, zlgmcu译)

第十讲　Cortex-M3架构和指令系统

§1嵌入式系统的应用领域

§2 ARM公司简介

§3　Cortex-M3内核

§4　指令系统

§5　嵌入式编程

§10.4 指令系统

一、 Thumb2指令集

二、9种寻址方式

三、6大类指令

四、4种伪指令

§10.4.1 Thumb-2指令集

1）CM3处理器指令的特点：是加载/存储型的，指令集只能处理寄存器的数据，而且处理结果都要放回寄存器，而对数据的访问要通过专门的加载/存储指令来完成；

2） CM3 处理器采用ARM v7-M 架构。它包括所有的16 位thumb 指令集和基本的32位thumb-2 指令集架构，不能执行ARM 指令；

3）Thumb-2 在thumb 指令集架构（ISA）上进行了大量的改进，它与thumb 相比，代码密度更高，并且通过使用16/32 位指令，提供更高的性能。

§10.4.2 9种寻址方式

一、寻址方式：

根据指令中地址码字段来寻找实际操作数地址的方式；

二、ARM处理器的9种寻址方式：

① 寄存器寻址
② 立刻寻址
③ 寄存器偏移寻址
④ 寄存器间接寻址
⑤ 基址寻址
⑥ 多寄存器寻址
⑦ 堆栈寻址
⑧ 块拷贝寻址
⑨ 相对寻址

§10.4.2 9种寻址方式

① 寄存器寻址

操作数在寄存器中，指令中的地址码字段指出的是寄存器编号，指令执行时直接取出寄存器中的数值进行操作，执行速度快。例如：

```
MOV  R1,R2      ;R1=R2
SUB  R0,R1,R2   ;R0=R1-R2
```

§10.4.2 9种寻址方式

② 立刻寻址

指令中的地址码字段是操作数，即数据包含在指令中，取出指令时也就取出可以立刻使用的操作数。例如：

```
MOV   R1,#0xff000      ;R1=0xff000
SUBS  R0,R0,#1         ;R0=R0-1
```

§10.4.2 9种寻址方式

③ 寄存器偏移寻址

ARM指令集特有的寻址方式。

第2个寄存器操作数在与第1个操作数结合之前，先进行移位操作。例如：

```
MOV  R0,R2，LSL,#3    ;R2左移3位
                      ;R0=R2
ANDS R1,R1,R2,LSL R3  ;R2左移R位
                      ;R1=R1 AND R2
```

§10.4.2　9种寻址方式

④　寄存器间接寻址

寄存器间接指令中的地址码字段指出的是寄存器编号，目标操作数在寄存器指定地址的存储单元中，即寄存器为操作数的地址指针。例如：

```
LDR  R1,[R2]     ;R1=[R2]
SUB  R1,R1,[R2]  ;R1=R1-[R2]
```

§10.4.2 9种寻址方式

⑤ 基址寻址

寄存器的内容与指令中给出的偏移量相加，形成操作数的有效地址。基址寻址用于访问基址附近的存储单元，常用于查表、数组操作、功能部件的寄存器访问等。例如：

```
LDR  R2,[R3,#0x0c]
STR  R1,[R0,#-4]       ;R0=R0-4,[R0]=R1
```

§10.4.2 9种寻址方式

⑥ 多寄存器寻址

多寄存器寻址允许一条指令中16寄存器中的部分或全部寄存器。例如：

```
LDMIA  R1!,{R2-R7,R12}
    ;将R1指向的单元中的数据读出到R2～R7，
     R12中，R1自动加1。
STMIA  R0!,{R2-R7,R12}
    ;将R2～R7，R12中的数据保存到R0所指的
     单元中，R0自动加1。
```

§10.4.2　9种寻址方式

⑦　堆栈寻址

堆栈寻址采用SP寄存器作为指向存储器的指针，有几种形式：

```
LDMFA, STMFA                ;满递增
LDMEA, STMEA                ;空递增
LDMFD, STMFD                ;满递减
LDMED, SP!,{R1-R7,LR}  ;空递减
        ;数据出栈，放入R1-R7,LR中
STMED SP!,{R1-R7,LR}
        ;将R1-R7,LR的值压入堆栈
```

§10.4.2 9种寻址方式

⑧ 块拷贝寻址

多寄存器数据传送指令，将一块数据在寄存器组和数据存储器之间传送。下列指令的功能时将R1-R7的数据保存到存储器中，但指针变化方式不同。存储器指针的变化

```
STMIA R0!,{R1-R7} ；在数据保存后增加
STMIB R0!,{R1-R7} ；在数据保存前增加
STMDA R0!,{R1-R7} ；在数据保存后减小
STMDB R0!,{R1-R7} ；在数据保存前减小
```

§10.4.2　9 种寻址方式

⑨　相对寻址

由程序计数器PC提供基地址,指令中的地址码段为偏移量,形成操作数的有效地址。

```
        BL  SUBR1      ;调用SUBR1子程序
        BEQ LOOP       ;相等则跳转到LOOP
LOOP  MOV R6,#1
        ...
SUBR1   ...
        RET
```

§10.4.3 6大类指令

① 跳转指令

② 数据处理指令

③ 加载/存储指令

④ 协处理指令

⑤ 程序状态寄存器访问指令

⑥ 异常产生指令

§10.4.3 6大类指令

① 跳转指令

直接向程序计数器PC写入跳转地址值

MOV LR,PC ;PC=LR

以下指令的前后的跳转范围为32MB：

B Lable

BNE Lable

BL Lable ；跳转之前把当前地址PC保存到；LR(R14)，用于子程序调用

BLX Lable ；从ARM指令跳转到Thumb；指令，保存PC到LR

BX Lable ；跳转到ARM或Thumb指令

§10.4.3 6大类指令

② 数据处理指令

- 数据传送指令

MOV R1,R0 ；R1=R0

MOVS R1,R0 ；S表示更新CPSR寄存器中的

；条件标志位

- 算术逻辑运算指令

ADD、ADC、ADCS、SUB、SBC、SUBS

RSB、ORR、EOR、MUL、MLA、SMULL、

SMLAL、UMULL、UMLAL

- 比较指令

CMP 比较、CMN 取反比较、TST 、TEQ

§10.4.3 6大类指令

③ 加载/存储指令

用于寄存器与存储器之间的数据传送。

LDR/LDRB/LDRH,STR/STRB/STRH

B:字节操作指令　　H:半字操作指令

LDM/STM(批量)

IA:每次传送后地址加1

IB:每次传送前地址加1

DA:每次传送后地址减1

DB:每次传送前地址减1

FD:满递减堆栈　　ED:空递减堆栈

FA:满递增堆栈　　EA:空递增堆栈

§10.4.3　6大类指令

④　协处理器指令

ARM处理器可支持最多16个协处理器，ARM协处理器指令有5条：

CDP：通知ARM协处理器执行特定的操作，若协处理器未能完成，则产生未定义指令异常

LDC：协处理器数据加载指令

STC：协处理器数据存储指令

MCR：ARM寄存器到协处理器寄存器的数据传送指令

MRC：协处理器寄存器到ARM处理器数据传送指令

§10.4.3　6大类指令

⑤ 程序状态寄存器访问指令

程序状态寄存器是指CPSR或SPSR。

MRS: 程序状态寄存器到通用寄存器的数据传送指令。

```
MRS  R0,CPSR     ; R0=CPSR
MRS  R0,SPSR     ; R0=SPSR
```

MSR :通用寄存器到程序状态寄存器的数据传送指令。

```
MSR  CPSR,R0     ; CPSR=R0
MSR  SPSR,R0     ; SPSR=R0
```

§10.4.3　6大类指令

⑥　异常产生指令

SWI: 软件中断指令

BKPT: 软件断点中断指令,用于程序调试

BKPT 16位立即数

§10.4.4 四种伪指令

一、伪指令

是编译器约定的指令，仅在汇编过程中起作用；不是CPU的指令，没有对应的操作码；

二、ARM的伪指令集共有4类

① 符号定义伪指令

② 数据定义伪指令

③ 汇编控制伪指令

④ 其它常用伪指令

§10.4.4 四种伪指令

① 符号伪指令

- GBLA/GBLL/GBLS ;定义全局变量

A:数字变量,默认值0； L:逻辑变量,默认值F；

S:字符串变量, 默认值""

- LCLA/LCLL/LCLS ;定义局部变量
- SETA/SETL/SETS ;变量赋值

```
LCLA    TEST3           ; 默认值0
TEST3   SETA    0X55 ;TEST3=55H
```

- RLIST: 定义寄存器的列表名称

```
Reglist  RLIST  {R0-R5,R8,R10}
```

将列表名称定义为Reglist,在ARM指令

LDM/STM中通过Reglist访问寄存器列表

§10.4.4 四种伪指令

② 数据定义伪指令

为特定的数据分配存储单元，完成存储单元初始化

- DC *：定义数据

  DCB/DCW/DCWU/DCD/DCDU

  DCFS/DCFSU/DCQ/DCQU　　U:非严格对齐

- SPACE: 分配一个连续的内存单元

```
DS  SPACE 100 ;分配100个字节的连续单元
```

- MAP: 定义一个结构化的内存表首地址
- FIELD: 定义一个结构化的内存表的数据域

```
MAP  0X100  ;定义结构化内存表首地址的值100H
A    FIELD  16  ;A的长度16字节，位置0X100
B    FIELD  32  ;B的长度为32字节，位置0X110
```

§10.4.4 四种伪指令

③ 汇编控制伪指令

用于控制汇编程序的执行流程。

- IF、ELSE、ENDIF； 可以嵌套使
- WHILE 逻辑表达式

(指令序列)

WEND

- MACRO、MEND:宏指令,可以嵌套使用

$ MACRO Var1,...VarN

(指令序列)

MEND ;用标号$代替指令序列

MEXIT ;用于从宏定义中跳转出去

§10.4.4 四种伪指令

④ 其它常用的伪指令

- AREA ：定义数据段、代码段
- ALIGN ：对齐方式
- ENTRY ：指定汇编程序入口
- END ：汇编程序结束
- EQU ：定义一个标号的值
- INPORT：通知编译器后面的标号在其它源文件中
- EXPORT：定义全局标号,供其它程序使用
- EXTERN：标号在其它源文件中,不用就不引用
- INCLUDE(GET)：源文件要包含的文件
- INCBIN：源文件要包含的数据文件,目标文件

第十讲　Cortex-M3架构和指令系统

§1嵌入式系统的应用领域

§2 ARM公司简介

§3　Cortex-M3内核

§4　指令系统

§5　嵌入式编程

§10.5 嵌入式编程

一、汇编语句格式

二、汇编程序结构

三、ATPCS规则

四、汇编和C的混合编程

§10.5.1 汇编语句格式

```
;文件名：TEST1.S
;功能：实现两个寄存器相加
;说明：使用ARMulate软件仿真调试
        AREA    Example1,CODE,READONLY    ;声明代码段Example1
        ENTRY                             ;标识程序入口
        CODE32                            ;声明32位ARM指令
START   MOV     R0,#0                     ;设置参数
        MOV     R1,#10
LOOP    BL      ADD_SUB                   ;调用子程序ADD_SUB
        B       LOOP                      ;跳转到LOOP
ADD_SUB
        ADDS    R0,R0,R1                  ;R0 = R0 + R1
        MOV     PC,LR                     ;子程序返回
        END                               ;文件结束
```

§10.5.1 汇编语句格式

① 格式

{标号} {指令或伪指令}　;{标注}

- 符号：区分大小写，自定义符号
  不能系统保留字相同
- 表达式和运算符

  +,-,X,/,MOD

  ROL,ROR,SHL,SHRAND,OR,NOT

  =,<,>,>=,<=,/=,<> 等

§10.5.2 汇编程序结构

一、可执行映像文件的组成

- 一个或多个代码段，代码段的属性为只读；
- 零个或多个包含初始化数据的数据段，数据段的属性为可读写；
- 零个或多个不包含初始化数据的数据段，数据段的属性为可读写。

二、汇编程序的基本结构

```
        PRESERVE8                ;//指示编译器8字节对齐
        THUMB                    ;//指示编译器为THUMB指令
        AREA    RESET, CODE,READONLY
START
        LDR   R0,=0x3FF6000  ;R0=3FF6000H
        ....
        END
```

§10.5.2 汇编程序结构

三、调用方法

通过BL指令来实现，保存返回地址到LR，再跳转。

```
        AREA    Init,CODE,READONLY
        ENTRY
Start   ....
        BL   sub1   ;调用子程序sub1，相当于call
        ...
  sub1   ...
        MOV  PC,LR  ;子程序返回，相当于return
        ...
        END
```

§10.5.3 ATPCS规则

① ATPCS--寄存器使用规则

- 子程序间通过寄存器R0-R3来传递参数，被调用的子程序在返回前无需恢复R0-R3的内容；
- 在子程序中，用寄存器R4-R11保存局部变量；
- 如果子程序使用其中的寄存器，在返回前必须恢复该寄存器的值，在Thumb中只能用R4-R7；
- R12用在过程调用的中间临时变量，记做IP；
- R13用作SP，子程序中不能用作其它用途，进入和退出时要保持不变；
- R14用作LP，用于子程序的返回地址，如果在子程序中保留了返回地址，则R14可以使用；
- R15用作PC，不能用作其它用途。

§10.5.3 ATPCS规则

② ATPCS-- 堆栈的使用规则

- ATPCS规定堆栈为FD类型，即满减堆栈，并且对堆栈的操作为8个字节对齐。
- 如果汇编程序在目标文件中包含外部调用，则必须满足下列条件：

  ---外部接口的堆栈为8个字节对齐；

  ---在汇编程序中使用PRESERVE8伪指令，告诉连接器，汇编程序数据是8字节对齐的。

§10.5.3 ATPCS规则

③ ATPCS-- 参数传递规则

- **参数个数可变的参数传递规则**

  当参数不超过4个时，通过寄存器R0-R3来传递参数；超过4个时，多余的参数使用堆栈来传递参数，压栈时最后一个数据先入栈。

- **参数个数固定的参数传递规则**

  参数依次保存到寄存器R0-R3，参数超过4个时，多余的参数使用堆栈来传递参数，压栈时最后一个数据先入栈，把所有参数当作连续存放的字数据。

§10.5.3 ATPCS规则

- 子程序结果返回规则

---结果为32位整数，通过寄存器R0返回；

---结果为64位整数，通过寄存器R0和R1返回；

---结果为浮点数，通过浮点运算部件的寄存器f0、d0或s0返回；

---结果为复合型浮点数，通过浮点运算部件的寄存器f0－fn、或 d0-dn返回；

---对于位数更多的结果，需要通过内存来传递。

§10.5.4 混合编程

一、在C/C++代码中嵌入汇编指令

```
_ASM
{ 内嵌的汇编指令
};
```

二、程序之间通过全局变量进行互访

全局变量在汇编程序中需要用IMPORT伪指令声明

三、ASM与C之间的调用须遵循ATPCS 规则

§10.5.4 混合编程

五、调用汇编的C程序例

① C程序代码

```
#include <stdio.h>
extern void strcopy(char *d,constchar *s)
;//要调用的汇编函数
int main(void)
{
    const char *srcstr="abc";
    char dststr;
    strcopy(char *d,constchar *s);//调用
    return(0);
}
```

§10.5.4 混合编程

五、调用汇编的C程序例

② 被调用的汇编程序

```
        AREA    SCopy,CODE,READONLY
        EXPORT  strcopy
strcopy LDRB    R2,[R1],#1   ;R1为源字符串
        STRB    R2,[R0],#1   ;R0为目标字符串
        ....
        MOV     PC,LR
        END
```

§10.5.4 混合编程

六、调用C的汇编程序例

① 汇编代码

```
EXPORT   CSUM5
         AREA    Eample,CODE,READONLY
         IMPORT   sum5
CSUM5   STMFD    SP!,{LR}    ;LR入栈
                              ;R0=a,R1=b,R2=c,R3=d,R4=e
         STR  R4,{SP,#-4}     ;第5个参数e通过堆栈传递
         BL       sum5        ;调用函数，结果在R0中
         ADD   SP,SP,#4       ;修正SP指针
         LDMFD   SP!,{PC}     ;子程序CSUM5返回
         END
```

§10.5.4 混合编程

六、调用C的汇编程序例

② 被调用的C程序

```
int sum5 ( int a, int b, int c, int d, int e)
{
    return(a+b+c);
}
```

谢谢！